

# Name Disambiguation from link data in a collaboration graph using temporal and topological features\*

Tanay Kumar Saha<sup>†</sup>, Baichuan Zhang<sup>†</sup> and  
Mohammad Al Hasan

the date of receipt and acceptance should be inserted later

**Abstract** In a social community, multiple persons may share the same name, phone number or some other identifying attributes. This, along with other phenomena, such as name abbreviation, name misspelling, and human error leads to erroneous aggregation of records of multiple persons under a single reference. Such mistakes affect the performance of document retrieval, web search, database integration, and more importantly, improper attribution of credit (or blame). The task of entity disambiguation partitions the records belonging to multiple persons with the objective that each decomposed partition is composed of records of a unique person. Existing solutions to this task use either biographical attributes, or auxiliary features that are collected from external sources, such as Wikipedia. However, for many scenarios, such auxiliary features are not available, or they are costly to obtain. Besides, the attempt of collecting biographical or external data sustains the risk of privacy violation. In this work, we propose a method for solving entity disambiguation task from link information obtained from a collaboration network. Our method is non-intrusive of privacy as it uses only the time-stamped graph topology of an anonymized network. Experimental results on two real-life academic collaboration networks show that the proposed method has satisfactory performance.

## 1 Introduction

On January 17, 2014, in his speech regarding the usages of phone surveillance data by NSA (National Security Agency), the USA President Barack Obama said, “This program does not involve the content of phone calls, or the names of people

---

\* A conference version of this paper is accepted in ASONAM 2014 conference, which is available for download from the last author’s webpage (<http://cs.iupui.edu/~alhasan/papers/anonymized-disambiguation.pdf>). The additional contributions in this journal version are explicitly listed in the appendix section of this paper. This research is supported by Mohammad Hasan’s NSF CAREER Award (IIS-1149851). † The first author and the second author contributed equally for this research

making calls. Instead, it provides a record of phone numbers and the times and lengths of calls—metadata that can be queried if and when we have a reasonable suspicion that a particular number is linked to a terrorist organization.” In this talk he also mentioned the importance of balancing security and privacy in all surveillance works of government agencies. However, making this balance is not an easy task; respecting privacy does not allow tapping into someone’s non-public biographical records; on the other hand, constrained analysis without detailed biographical data leads to numerous false identification and entity mixup. In this work, we are concerned with solving the task of entity disambiguation without using biographical information—the input to our solution is link data collected from anonymized collaboration networks, similar to the one that Mr. Obama has explained.

Entity disambiguation [7, 21] is not a new problem. In fact, named entity disambiguation has been a long standing problem in the field of bibliometrics and library science. The key reason for this is that many distinct authors share the same name, specifically considering the fact that the first name of an author is typically written in abbreviated form in the citation of many scientific journals. Thus, bibliographic servers that maintain such data may mistakenly aggregate the records from different persons into a single entity. For example consider DBLP, the largest bibliographic website of computer science. In DBLP, there are at least 8 distinct persons named Rakesh Kumar, and their publications are mixed in the retrieved citations. The ambiguity in Chinese names is more severe, as many Chinese share a few family names such as Wang, Li, and Zhang. An extreme example is Wei Wang. According to our labeling, it corresponds to over 200 distinct authors in DBLP! To correctly assess the impact of a researcher in a research field, correct attribution of research works is essential, so entity disambiguation has been extensively addressed by researchers in information retrieval and data mining. Note that, a related problem considers the task of merging multiple name references into a single entity, where the records belonging to a single person has been erroneously partitioned into multiple name references [2, 3, 20, 27, 28]. This task is more popularly known as entity deduplication or record linkage, and it is not the focus of this work.

Many research works are proposed to solve the entity disambiguation. Among those, some are specifically targeted for solving the name entity disambiguation [4, 7–9, 18]. Existing works mostly use biographical features, such as name, address, institutional affiliation, email address, and homepage; contextual features, such as coauthor/collaborator, and research keywords; and external data such as Wikipedia [7]. From methodological point of view, some of the works follow a supervised learning approach [8, 10], while others use unsupervised clustering [5, 9, 17, 25]. There exist quite a few solutions that use graphical models [3, 23, 26, 31]. What is common among all these works is that they use many biographical features including name, and affiliation, so they cannot protect the privacy of the actors in the dataset. Using biographical features is acceptable for entity disambiguation of authors in the field of bibliometrics, but it raises a serious concern when it is being used for applications related to national security; in such an application, it is more desirable to use a pre-filter that identifies a small list of suspicious data references for which biographical data can be queried after the approval of a privacy management officer. Besides the concern of privacy, significant cost is also involved in collecting biographical information, which

impedes the effective utilization of the existing methodologies for solving entity disambiguation.

In this work, we consider an anonymized social network. Each node in this network corresponds to a reference to a name entity, and each edge corresponds to collaboration among different name entities. The edges are labeled with time-stamps representing the time when a collaboration took place. As we have discussed earlier, we can think of such a network as the anonymized email/communication network that the NSA uses to identify suspects. Our solution to entity disambiguation in an anonymized network uses the timestamped network topology around a vertex of the network and by using an unsupervised method it produces a real-valued score for that vertex. This score represents the degree to which a given anonymized reference (a vertex) is pure. The smaller the score, the more likely that the reference may comprise of records of multiple real-life entities. For a given vertex, the method provides the desired score in a few seconds, so one can always use it as a pre-filter to identify a small set of target nodes for which more thorough analysis can be made subsequently. Alternative to an unsupervised approach, our method can also be adapted to a supervised classification system for predicting the purity status of a node, when a labeled dataset is available.

We claim the following contributions in this work:

- We design the task of solving name entity disambiguation using only graph topological information. This work is motivated by the growing need of data analysis without violating the privacy of the actors in a social network.
- We propose a simple solution that is robust and it takes only a few seconds to disambiguate a given node in real-life academic collaboration networks. The proposed method returns a real-valued score to rank the vertices of a network based on their likelihood for being an ambiguous node. So the score can be used as a pre-filter for identifying a small set of ambiguous references for subsequent analysis with a full set of features. Besides, the score can also be used independently as a feature for classification based solutions for entity disambiguation.
- We use two real-life datasets for evaluating the performance of our solution. The results show that the method performs satisfactorily, considering the fact that it uses only the topology of a node in its analysis.

## 2 Related Work

In existing works, name disambiguation task is studied for various entities; examples include disambiguation on Encyclopedic knowledge or Wikipedia Data [4, 7, 13], citation data [8, 9, 22, 29], email data [18], and text documents [14, 21].

In terms of methodologies, both supervised [4, 8] and unsupervised [9] approaches are considered. For supervised method, a distinct entity can be considered as a class, and the objective is to classify each event to one of the classes. Han et al. [9] use such a framework, and propose two supervised methods, one using a generative model, and the other using SVM. In another supervised approach, Bunescu et al. [4] solve name disambiguation by designing and training a disambiguation SVM kernel that exploits the high coverage and rich structure of the knowledge encoded in an online encyclopedia. However, the main drawback

of supervised methods is the lack of scalability, and the unavailability of labeled data for training. It is also impractical to train thousands of models, one for each individual, in a large digital library.

For unsupervised name disambiguation, the collaboration events are partitioned into several clusters with a goal that each cluster contains the events corresponding to a unique entity. Han et al. [9] propose one of the earliest unsupervised name disambiguation methods, which is based on  $K$ -way spectral clustering. They apply their method for name disambiguation in an academic citation network. For each name dataset, they calculate a Gram matrix representing similarities between different citations and apply  $K$ -way spectral clustering algorithm on the Gram matrix to obtain the desired clusters of the citations. In another unsupervised approach, Cen et al. [5] compute pairwise similarity for publication events that share the same author name string (ANS) and then use a novel hierarchical agglomerative clustering with adaptive stopping criterion (HACASC) to partition the publications in different author clusters. Malin [17] proposes another cluster-based method that uses social network structure.

Probabilistic relational models, specifically graphical models have also been used for solving entity disambiguation task. For example, authors in [31] propose a constraint-based probabilistic model for semi-supervised name disambiguation using hidden Markov random fields (HMRF). They define six types of constraints and employ EM algorithm to learn the HMRF model parameters. In another work, Tang et al. [23, 26] present two name disambiguation methods that are based on pairwise factor graph model. They target name disambiguation in academic datasets. In their work, the authorship of a paper is modeled as edges between observation variables (papers) and hidden variables (author labels). Features of each paper and relationships, such as, co-publication-venue and co-author, have impact on the probability of each assignment of labels. The similarity between two clusters is encoded in different factors (edge potentials) on different features. The clustering process iterates over different author label assignments and selects the one with maximal probability. An improved version of the above work uses user feedback and is being used in the real-life Arnetminer system ([arnetminer.org](http://arnetminer.org)) for disambiguation [26]. LDA based context-aware topic models has also been used for entity disambiguation [21].

To build the features for classification, clustering, or probabilistic models, most of the existing works use biographical and contextual attributes of the entities or external sources such as Wikipedia, Web search results and online encyclopedia. In almost every work on name disambiguation, person’s name, email address, and institutional affiliation are used. It is not surprising, because biographical features are highly effective for name disambiguation. For instance, a set of recent works [6, 15] report around 99% accuracy on a data mining challenge dataset prepared by Microsoft research. These works use supervised setup with many biographical features; for instance, one of the above works even predict whether the author is Chinese or not, so that more customized model can be applied for these cases.

In this work, we consider the task of name disambiguation using only the network topological features in an anonymized collaboration graph. We found a recent work [10] which also has a similar objective. They consider this problem in the supervised setting where they are provided with a base graph and a set of nodes labeled as ambiguous or unambiguous. They characterize the similarity between two nodes based on their local neighborhood structure using graph kernels and

solve the resulting classification task using SVM. We will show in the experiment section that our method performs significantly better than this method in terms of both speed and accuracy. Note that, a preliminary version of our paper is already published as a short conference article [30].

### 3 Solution Overview

We assume that a collaboration network  $G(V, E)$  is given, where each node  $u \in V$  represents an entity reference which in real-life may be linked to multiple persons. For every edge  $e \in E$  we are also provided with a list  $T(e)$  which represents the discrete time-point at which the collaboration events between the corresponding nodes have taken place. Our objective is to predict how likely it is that the node  $u$  is a multi-node, i.e., it comprises of collaboration records of multiple persons. We use a linear model to produce a numeric score  $s(u)$ , which represents the likelihood that  $u$  is a pure node.

To solve the problem in the setup discussed in the last paragraph, we first construct the ego network of  $u$ ,  $G_u \subseteq G$ , which is an induced subgraph of  $G$  consisting of ego node  $u$  and all of its direct neighbors (these nodes are called “alters”). Since  $G_u$  is an induced subgraph, it preserves the ties between  $u$  and the alters and also the ties between a pair of alters. We hypothesize that if  $u$  is a multi-node, the graph  $G_u$  will form many disjoint clusters, once the node  $u$  and all of its incident edges from  $G_u$  are removed; each of these clusters corresponds to one of the many real-life entities that have been merged together under the reference  $u$ . This hypothesis is built from the transitivity property of a social network, which states that the friends of your friend have high likelihood to be friends themselves [11]. Thus, if  $v$  and  $w$  are friends of  $u$ , with high likelihood, there are edges between  $v$  and  $w$ . However, when  $u$  is a multi-node corresponding to  $k$  different people, the friends of  $u$  are partitioned into at least  $k$  disjoint clusters.

In Figure 1, we illustrate our hypothesis. Assume that the triangle shaped node is  $u$ , and the graph in this figure corresponds to the ego network of  $u$ ,  $G_u$ . We also assume that  $u$  is a multi-node consisting of two name entities. So the removal of the node  $u$  (along with all of its incident edges) from  $G_u$  makes two disjoint clusters; this phenomenon is illustrated in the lower part of figure 1. The vertices of these clusters are shown using circles and squares respectively.

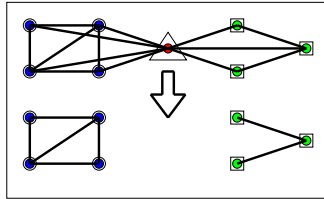


Fig. 1: A toy example of clustering based entity disambiguation

However, there are several caveats in the above simplified formulation. Particularly, the above formulation may yield numerous false alarms for various reasons even if a node  $u$  is not a multi-node. First, if the name entity at  $u$  participates

in multiple communities, her neighbors may form disjoint clusters. Second, the neighborhood of the entity  $u$  may have several distinct clusters considering the temporal axis, which happens if  $u$  changes job, institution or affiliation. False negatives also occur, though with a lesser likelihood. For a multi-node  $u$ , if the significant parts of the collaboration activities of  $u$  comprise of only one name entity, the remaining weaker entities under  $u$  contribute poorly in the score  $s(u)$ , which may prevent from categorizing  $u$  as a multi-node. Another challenge is that due to the power-law behavior of a typical collaboration graph  $G$ , the neighborhood graph  $G_u$  has varying size and density, which affects the comparison of the score value  $s(\cdot)$  of various nodes in  $G$ . We take into consideration each of these problems in our proposed solution, as explained in next sections.

#### 4 Methods

We assume that a collaboration network  $G = (V, E)$  and a ego node  $u$  is given, where  $u \in V$  represents an entity reference, which in a real-life scenario may be linked to multiple persons. From  $G$  and  $u$ , we construct the ego network of  $u$ ,  $G_u = (V_u, E_u) \subseteq G$ . For each edge in  $G_u$ , say  $e_u = (v, w) \in E_u$ ,  $T(e_u)$  represents the set of collaboration events between  $v$  and  $w$  that are captured by  $G_u$ ; i. e.  $T(e_u) = \{\langle n_i, t_i \rangle\}_{1 \leq i \leq |T|}$ , here,  $n_i$  is the number of collaboration events at time  $t_i$ . From  $T(e_u)$ , we compute a similarity value between  $v$  and  $w$  under  $G_u$  using an exponential decay function as shown below:

$$W_{G_u}(v, w) = \sum_{i=1}^{|T|} n_i \times \exp \frac{-(t_{\max} - t_i)}{\tau}$$

In the above equation,  $t_{\max}$  denotes the most recent time when a collaboration event happened between any two vertices in  $G_u$ .  $\tau$  is a tuning parameter that one can use to control the rate of the decay. On academic collaboration networks, for which the time unit is year, we use  $\tau$  ranging between 5 to 10. Empirical results on such networks also show that the performance of the system is not sensitive to the choice of  $\tau$ . The above definition of edge similarity function rewards more weight to recent collaboration events than to older collaboration events.

**Example:** Given  $G_u = (V_u, E_u)$ ;  $v, w \in V_u$ . Let us assume  $t_{\max} = 2014$ . Besides,  $v$  and  $w$  has 2 collaboration events in 2014, 3 collaboration events in 2013 and 4 collaboration events in 2010; thus,  $T((v, w)) = \{\langle 2, 2014 \rangle, \langle 3, 2013 \rangle, \langle 4, 2010 \rangle\}$ . By setting  $\tau = 5$ , we get,  $W_{G_u}(v, w) = 2 \times \exp \frac{-(2014-2014)}{5} + 3 \times \exp \frac{-(2014-2013)}{5} + 4 \times \exp \frac{-(2014-2010)}{5} \approx 6.25$ .  $\square$

##### 4.1 Obtaining Cluster Quality Score

Given the collaboration network,  $G$  and a specific vertex  $u$ , the next step of our method is to construct  $G_u$ —the ego network of  $u$ . Then, we cluster the graph  $G_u \setminus \{u\}$  using a graph clustering algorithm. The objective of this clustering is to group  $u$ 's neighbors in different clusters such that the cross-edges between different

clusters are minimized. The reason of removing  $u$  is to find whether the neighbors of  $u$  are strongly connected by themselves without using  $u$  as an intermediate vertex.

We utilize Markov Clustering (MCL) for the clustering task. MCL uses the graph’s natural transition probability matrix to cluster a graph by combining random walks with two alternating operations (expansion and inflation) [24]. There are several reasons for the choice of MCL. First, MCL is one of the fastest graph clustering methods; competing other methods, such as, spectral clustering and its variants [16] compute eigenvectors of graph Laplacian, which could be costly for large graphs. For our work, we found that for a graph with several thousands vertices, MCL finishes with good clustering results only within a few seconds. Second, MCL does not require the number of clusters as one of the input parameters, which works well for our setting as we have no information regarding the number of communities in which the node  $u$  participates. Finally, MCL is robust against the choice of parameters. It has only one parameter, called *inflation*, which we set to the default value in all of our experiments.

#### 4.1.1 Normalized Cut based Score

For our task, we are mainly interested in obtaining a score reflecting the quality of clustering. There are various evaluation criteria for clustering, including ratio cut, normalized cut [16] and modularity [19]. Among these, we choose the normalized cut based clustering score as it reflects the ratio of the similarity weight-sum of the inter-cluster edges and the same for all the edges in the graph. The equation of normalized cut score for a node  $u$  is shown below:

$$NC = \sum_{i=1}^k \frac{W(C_i, \overline{C_i})}{W(C_i, C_i) + W(C_i, \overline{C_i})} \quad (1)$$

where  $W(C_i, C_i)$  denotes the sum of weights of all internal edges and  $W(C_i, \overline{C_i})$  is the sum of weights for all the external edges and  $k$  is the number of clusters in the graph. We compute  $NC$  value by independent calculation after we obtain the clusters of  $G_u$  using MCL algorithm.

For a node  $u$ , the normalized cut (NC) score denotes the clustering tendency of the neighbors of  $u$ . If the value is high, then the clustering tendency of  $u$ ’s neighbors is poor, so  $u$  is less likely to be a multi-node. On the other hand, if the value is small, then  $u$ ’s neighbors are well clustered and  $u$  has a high probability to be a multi-node. For example, in the bibliographic domain, if a multi-node  $u$  in a co-authorship network represents two researchers who share the same name, with a high likelihood, they will have a distinct set of co-authors. After clustering the graph  $G_u \setminus \{u\}$  using MCL, we expect to obtain two dominant clusters that are disconnected (or very sparsely connected), each representing the set of co-authors of each of the two researchers. One problem with the  $NC$ -score is that it is not size invariant, i. e., if a node  $u$  has many clusters, its  $NC$ -score is large, so this score is not that useful when we compare the  $NC$ -scores of many nodes to find the one which is likely to be a multi-node. In order to address this issue, we normalize the score by the number of clusters as shown below:

$$NC\text{-score} = \frac{NC}{k} \quad (2)$$

where  $k$  is the total number of clusters that we obtain using MCL, given the ego network of  $u$ ,  $G_u$ .

#### 4.2 Obtaining Temporal Mobility Score

NC-score is a good metric to represent the degree at which a given anonymized entity is pure. However, for many real-life datasets, this score yields many false positives. Assume, a vertex in a social network represents one real-life entity, but its collaboration network evolves over time because of entity mobility. For such a vertex, the NC-score is small due to disjoint clusters of neighbors that are formed as the entity moves along with the time; this leads to a false positive prediction that the entity is likely to be a multi-node. Since the anonymized collaboration data has the time stamp of the collaboration event, we use this information to obtain a second score that we call Temporal Mobility (TM) score. This score indicates how likely is that the specific entity has moved along with the time. Note that, temporal mobility is not a new concept, it has been studied by social scientists earlier; for instance, there are works to understand the temporal mobility of academic scientists as they change their jobs [1].

To compute the likelihood that a given vertex,  $u$ , has experienced temporal mobility, we obtain the *TM*-score (temporal mobility score) of  $u$ . This score reflects the extent by which the node  $u$  collaborates with different neighbor clusters (obtained using MCL) at a distinct time range. Below we discuss the process that computes the *TM*-score of a node  $u$ .

Let's assume that  $C_{i:1 \leq i \leq k}$  are  $k$  different clusters that we obtain from MCL during the cluster quality computation stage. To model  $u$ 's collaboration with a cluster  $C_i$ , we first obtain a vector,  $Z_i$  of size  $|T|$  (the number of distinct time intervals in  $u$ 's collaboration history), in which each entry denotes the total number of collaboration events between  $u$  and the members of  $C_i$  at that specific time interval. If a collaboration event with  $u$  consists of  $l (\geq 2)$  actors (besides,  $u$ ), each of the actors in cluster  $C_i$  contributes  $1/l$  to the appropriate entry in  $Z_i$ . Say,  $u$  publishes a paper with  $l$  co-authors at year  $t$ . After clustering,  $m$  of the co-authors out of the  $l$  co-authors of that paper belong to the cluster  $C_i$ , this event additively contributes to  $Z_i$  vector's  $t$ 'th position by  $m/l$ . Thus for a multi-party events, a collaboration event with  $u$  can be distributed among different clusters, in case  $u$ 's accomplices for this event are distributed among different clusters. Thus, by iterating over all the collaboration events of  $u$ , we compute  $k$  different vectors,  $Z_i : 1 \leq i \leq k$ , each such vector corresponds to  $u$ 's collaboration with the entities in one of the clusters,  $C_i$ . Finally, we use centered moving average to smooth the  $Z_i$  vectors.

For the purpose of illustration, we present an example of temporal mobility of an entity in Figure 2. In this Figure we can observe 3 histograms; each histogram shows the yearly count of collaboration events that this entity has with other entities in one of his neighbor-clusters. We also present the same information after smoothing with the centered moving average. We draw the histograms in this figure using real-life data of a researcher in DBLP academic collaboration network. In the DBLP data, all the authorship records of this researcher point to a single name reference. However, when we use MCL clustering, we obtain 3 dominant clusters for this person, with almost no inter-cluster edges leading to a small NC-score. This suggests that this entity has a high chance to be a multi-node. But, as



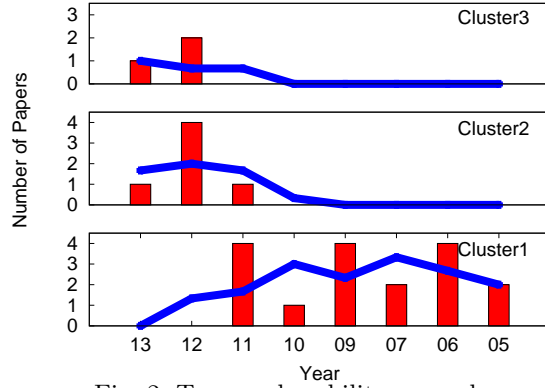


Fig. 2: Temporal mobility example

shown in Figure 2, the association pattern of this entity with the three different clusters suggests temporal mobility. During 2005-2010 time period, the entity has almost dedicated association with the entities in Cluster 1; which is followed by a divided association among the entities in Cluster 2 and Cluster 3 for the time period 2011-2013.

The name of the entity that we discuss here is Dr. Honglak Lee. The first cluster corresponds to his collaboration with his co-authors when he was a PhD student in the Stanford University. The second cluster denotes his collaboration with his PhD students at the University of Michigan. The third cluster represents his collaboration with his colleagues in the same university. This example illustrates that how temporal mobility of an entity can lead to a false positive multi-node when only NC-score is used for this qualification.

#### 4.2.1 Temporal Mobility score using KL Divergence

Our main objective in this step is to obtain the Temporal Mobility (TM) score of a node  $u$  after we obtain  $u$ 's cluster-wise collaboration vectors,  $Z_i$ . For this we convert each of the smoothed  $Z_i$  vectors to a discrete probability distribution by normalizing these vectors appropriately. Thus, each of  $u$ 's neighbor-clusters are represented by a discrete probability vector. To compute the temporal divergence of these clusters we use Kullback-Leiber (KL) divergence which is a measure of divergence between two probability distributions.  $D(P \parallel Q)$  denotes the KL divergence between two probability distributions  $P$ ,  $Q$ , and it is defined on a finite set  $\chi$  as below:

$$D(P \parallel Q) = \sum_{x \in \chi} P(x) \log \frac{P(x)}{Q(x)}$$

This value is large, if the two distributions are different, and vice-versa. Note that,  $D(P \parallel Q)$  is an asymmetric metric. So for our task we use symmetric KL divergence which for the distributions  $P$  and  $Q$  are  $D(P \parallel Q) + D(Q \parallel P)$ . Also, to avoid the scenario when the discrete distributions of  $P$  and  $Q$  contain a zero element, we adopt Laplace correction that assigns a small probability (0.01) to those entries.

Now,  $TM$ -score of  $u$  is simply the weighted average value of the symmetric KL divergence of all pairs of  $Z_i$  (normalized to 1) vectors. For each pair,  $Z_i$ , and  $Z_j$ , we first compute  $D(Z_i \parallel Z_j)$  and  $D(Z_j \parallel Z_i)$ . The weight of the divergence between  $Z_i$  and  $Z_j$ , is denoted as  $w(Z_i, Z_j)$ ; we use the sum of the number of events in the cluster  $C_i$  and  $C_j$  as this weight. Using such weighting, the KL-divergence between dominant clusters contribute more in the  $TM$ -score. Similar to the case of  $NC$ -score, we also normalize  $TM$ -score by the number of clusters, so that different nodes with diverse number of clusters can be compared. The overall computation can be shown using the following equation:

$$TM\text{-score} = \frac{\sum_{i=1}^{k-1} \sum_{j=i+1}^k w(Z_i, Z_j) \cdot \left( D(Z_i \parallel Z_j) + D(Z_j \parallel Z_i) \right)}{k \times \sum_{i=1}^{k-1} \sum_{j=i+1}^k w(Z_i, Z_j)} \quad (3)$$

The higher the value of  $TM$ -score of an entity, the higher the likelihood that the node is not a multi-node. Rather it has experienced the temporal mobility phenomenon along its overall time intervals.

#### 4.3 Linear Model using $NC$ -score and $TM$ -Score

We can use  $NC$ -score and  $TM$ -score for unsupervised learning. For an unsupervised case, we simply predict a score  $s(u)$  for a node  $u$ . The higher the score, the more likely that the node is a pure node. For this we use a linear model with only one model parameter  $\alpha$ , which is positive, because both  $NC$ -score and  $TM$ -score have larger values for a pure node and smaller value for a multi-node. Thus, the score  $s(u)$  of a node  $u$  is simply:

$$s(u) = NC\text{-score}(u) + \alpha \cdot TM\text{-score}(u) \quad (4)$$

The model parameter  $\alpha$  should be set manually depending on the nature of the dataset. In co-authorship networks, a small  $\alpha$  value in the range from 0.1 to 0.2 works well. The benefit of this unsupervised method is that we can simply work on a small collection of nodes independently. By sorting the  $s$ -score of those nodes in increasing order, we can identify the top-ranked nodes that are likely to be suspected of being a multi-node.

---

#### Algorithm 1 Unsupervised-Disambiguation( $G, u$ )

---

**Input:**  $G, u$

**Output:**  $s(u)$

- 1: Construct the ego network  $u$ ,  $G_u$  using the similarity weight between vertices
  - 2: Remove  $u$  from  $G_u$  and apply MCL to get  $k$  clusters  $\{C_i\}_{1 \leq i \leq k}$
  - 3: Using Equation 1 and Equation 2, compute  $NC$ -score
  - 4: For each cluster  $C_i$ , compute normalized  $Z_i$  vector using timestamp of association
  - 5: Use Equation 3 to compute  $TM$ -score
  - 6: Using a set of validation dataset, tune the  $\alpha$  value for the given dataset
  - 7: **return**  $s(u) = NC\text{-score}(u) + \alpha \cdot TM\text{-score}(u)$
-

#### 4.4 Pseudo-code and Complexity Analysis

The pseudo-code of the entire process for the unsupervised setup is given in Algorithm 1. It takes an input graph  $G$  and a specific ego node  $u$  as input and generates the numeric score of  $u$ ,  $s(u)$  as output. Line 1 computes the similarity weights for the edges of the ego network of  $u$ ,  $G_u$ . Line 2 removes  $u$  from  $G_u$ , and applies MCL clustering method to cluster the similarity graph  $G_u$ . We assume that this clustering yields  $k$  clusters,  $\{C_i\}_{1 \leq i \leq k}$ . From these clusters, Line 3 obtains the normalized cut based score. For each cluster (say,  $C_i$ ), Line 4 computes the temporal collaboration vector  $Z_i$  of each cluster which represents the association weight between the entities in a cluster and  $u$  over the time axis. Line 5 obtains the  $TM$ -score using temporal mobility model that we discuss in Section 4.1. Line 6 tunes the model parameter  $\alpha$  for the given dataset and Line 7 returns the desired score for the node  $u$ . A high value of this score is more likely to represent a pure node and a small value makes the node more likely to be a multi-node. Thus our method can be used as a pre-filter to identify a small set of nodes that are more likely candidate for being a multi-node.

Given the collaboration network  $G$  and a specific vertex  $u$  as input, generation of ego network  $G_u(V_u, E_u)$  takes  $\mathcal{O}(|V_u| + |E_u|)$  time. The computational complexity of MCL algorithm is  $\mathcal{O}(t \cdot |V_u|^3)$  in the worst case, where  $t$  is the number of iterations until convergence. The steps in Line 3 and Line 4 read the similarity matrix of  $G_u$  using an adjacency list representation; thus the complexity of these steps is roughly  $\mathcal{O}(|V_u| + |E_u|)$ . The KL divergence computation in Line 5 uses Equation 3 to compute  $TM$ -score, which has a cost of  $\mathcal{O}(cK^2)$ , where  $K$  is the number of clusters after MCL clustering. Thus, the overall time complexity of Algorithm 1 is  $\mathcal{O}(|V_u|^3)$ . However, note that the above complexity bound is over the size of the ego network instead of the entire collaboration network  $G$ , which makes the proposed method very efficient. We also present the running time of our method in 5.6 over a set of real-life networks.

#### 4.5 Supervised Classification Setup

In a supervised classification setting, we can use the  $NC$ -score and the  $TM$ -score as classification features. For this, we first build a training dataset in which the exact labels (positive for a multi-node, negative otherwise) of each of the instances are known. Then we can use any of our favorite classification methods to build a model, which can later be used for predicting the label for an unknown data instance. Supervised classification setup enables adding of more features for the classification task. So, in this setup, we also consider centrality-based graph topology features, in addition to the  $NC$ -score and  $TM$ -score.

We consider four distinct centrality-based features [12]: degree centrality, betweenness centrality, closeness centrality, and eigenvector centrality. For a given node,  $u$  one can always compute the centrality values of  $u$  considering the entire collaboration network, however such a method is not scalable, so we compute  $u$ 's centrality within its ego-network,  $G_u$ . In  $G_u$ ,  $u$  is the central node by construction, but, more so, if  $u$  is a multi-node. This is because when multi-node,  $u$  naturally has a higher than usual degree in  $G_u$  leading to a high degree centrality value for  $u$ . Also, when  $u$  is a multi node,  $G_u$  is composed of many disjoint clusters and shortest paths among the nodes in distinct clusters must go through  $u$ , leading to

a high betweenness centrality for  $u$ . Similar arguments can also be made for other centrality metrics. However, one potential issue of computing centrality within  $G_u$  is that for different nodes, their centrality values in their respective ego networks are not comparable, so we normalize  $u$ 's centrality score over the centrality score of all nodes in  $G_u$  as below:

$$\text{Centrality-Score}(u) = \frac{C(u)}{\sum_{v \in G_u} C(v)}; \quad (5)$$

where,  $C(x)$  is the centrality value of a node  $x$  in  $G_u$ . We will show in experimental results that considering centrality metrics as feature improves the prediction performance.

## 5 Experiments and Results

A key challenge of working on the name entity disambiguation task is to find a real-life labeled dataset for evaluating the performance of the proposed solution. There exist real-life collaboration datasets, such as email or phone networks, but they are anonymized for security concern, so we can't really obtain the true ambiguity label of the nodes in such networks. Hence, these datasets are not useful for evaluating the entity disambiguation task. For our experiments, we use two well known bibliographic datasets, DBLP<sup>1</sup> and Arnetminer<sup>2</sup>. Both of these datasets are leading repositories for bibliographic information of computer science conferences and journals. Also, the ambiguity label of a scientist in any of these networks can be determined by manual inspection of the papers published by that scientist.

From both datasets, we select 150 researchers such that half of them are pure nodes (negative cases), and the rest of them are multi-nodes (positive cases). We try to make the datasets as representative as possible by choosing a mix of senior and early career researchers. To assign the label for a selected researcher, we manually examine her bibliographic records and also her webpage profile. Besides, in DBLP, name disambiguation ground truth is already available for a few of the high profile researchers. We use those ground truths to double-check our manual labeling. The final dataset is anonymized by mapping each researcher to a unique id.

The objective of our experiments with these datasets is to verify whether our method can distinguish the set of multi-nodes from the pure nodes. For this validation we use both supervised and unsupervised methodologies. We also compare our method with the existing state-of-the-art to show that our method is superior than that both in terms of speed and accuracy. Besides these, we also perform experiments to analyze the sensitivity of our method as the parameters vary.

Our method has only a few parameters, many of which we keep fixed. The first,  $\tau$ , denotes the exponential decay rate, which is used while computing the similarity between a pair of entities in the input network before the clustering step. We fix the  $\tau$  value to 5 for all of our experiments since the performance remains stable as we vary  $\tau$  for both the datasets. For the clustering of the collaboration network, we use MCL clustering method. We use the code provided by the inventor of MCL and set the inflation value of MCL to 1.4 (as is recommended by the inventor) for

<sup>1</sup> <http://dblp.org/search/index.php>

<sup>2</sup> <http://arnetminer.org>

all of our experiments. For the other data processing, we write our own code using Python. The experiments are performed on a 2.1 GHz laptop with 4GB memory running Linux operating system.

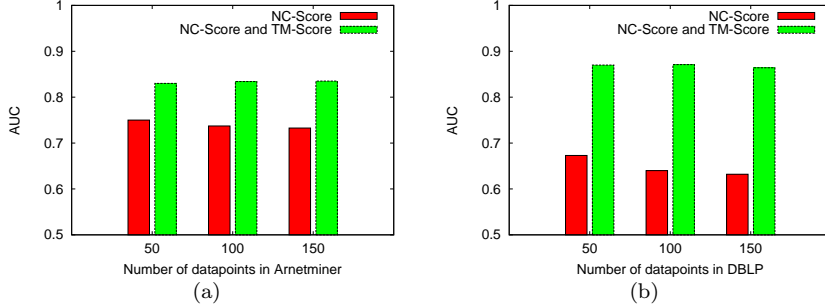


Fig. 3: Unsupervised disambiguation experimental results: (a) on Arnetminer and (b) on DBLP

### 5.1 Evaluation of Unsupervised Disambiguation

In unsupervised disambiguation, we do not train a model using a training dataset, rather we use a linear function (Equation 4) to obtain the  $s$ -score. For evaluating the performance of unsupervised disambiguation, in this experiment we compute the  $s$ -score of each of the 150 researchers in the DBLP and Arnetminer datasets using Equation 4. We use an  $\alpha$  value of 0.1 for Arnetminer and 0.2 for the DBLP dataset. The choice of  $\alpha$  is fixed by comparing the performance of our method on a small validation dataset by varying  $\alpha$  between 0 and 1. We use AUC (the area under the ROC curve) as the evaluation metric of this experiment which we obtain as below.

We sort the  $s$ -score of the 150 researchers in an increasing order and use each of the  $s$ -scores (in that order) as the threshold of our prediction to obtain a sequence of TPR (true positive rate) and FPR (false positive rate) pairs. From these (TPR, FPR) datapoints, we draw the ROC curve and subsequently compute the AUC value of our prediction. In this case, the AUC value is essentially the probability that the  $s$ -score of a random multi-node (positive data instance) is smaller than the  $s$ -score of a random negative data instance. The baseline value for the AUC is 0.5 and the best value of AUC is 1. The former case happens if the  $s$ -scores of positive and negative instances are non-distinguishable; on the other hand, the latter case happens when all the  $s$ -scores of the positive instances are smaller than all the  $s$ -scores of the negative instances. For the DBLP dataset and the Arnetminer dataset, the AUC value that our method achieves is 0.86 and 0.83 respectively, which, for AUC, is generally considered as excellent.

To show the variance of performance for inputs of different sizes, we construct 2 additional datasets, which are uniformly chosen random subset of the original dataset. These two datasets have 50, and 100 data instances respectively. For these datasets, we compute the AUC value as we described above. We repeat the above random dataset creation process for ten times and compute the average AUC

value for both datasets. We show the AUC comparison among these datasets in Figure 3(a) and 3(b) for the cases of Arnetminer and DBLP, respectively. As we can see for the Arnetminer case, the AUC value is almost constant (0.83) for all the three datasets with varying sizes. For DBLP, the datasets with 50 and 100 instances achieve an AUC value of 0.87, whereas the entire dataset with 150 instances achieves an AUC value of 0.86.

Method	Kernel	DBLP	Arnetminer
Our method using NC and TM features	Linear	72.50	65.60
	Radial basis	72.31	68.82
	Sigmoid	71.90	66.20
Our method using NC, TM and Graph Centrality features	Linear	75.60	67.00
	Radial basis	74.71	69.42
	Sigmoid	75.30	70.03
Method proposed in [10]	GL-3	40.67	43.62
	GL-4	41.33	44.98
	SP	48.22	47.67

Table 1: Comparison between our method and [10] using Classification accuracy (%) on 10-fold cross-validation

Method	Kernel type	DBLP	Arnetminer
Our method using NC and TM features	Linear	0.80	0.76
	Radial basis	0.79	0.75
	Sigmoid	0.79	0.75
Our method using NC, TM and Graph Centrality features	Linear	0.83	0.80
	Radial basis	0.82	0.79
	Sigmoid	0.80	0.78
Method proposed in [10]	SP	0.62	0.61
	GL-3	0.63	0.62
	GL-4	0.64	0.62

Table 2: Comparison between our method and [10] using AUC on 10-fold cross-validation

In Figure 3(a) and Figure 3(b) we also show experimental results that highlight the contribution of *TM*-score in our model. For this we compare the AUC value that we obtain using *TM*-score and without using *TM*-score; the second case can be obtained by setting  $\alpha=0$  in Equation 4. The AUC value of these two cases for dataset of different sizes are shown using green (dotted box) and red (solid box) bar plots, respectively. As we can see, for both datasets *TM*-score significantly improves the AUC score for the cases of all different sizes. For DBLP, the improvement is particularly significant; for the entire dataset (150 instances), the AUC without and with *TM*-score is 0.63 and 0.86 respectively. We guess that the reason for such dramatic improvement using *TM*-score is due to the fact that we use academic collaboration datasets, in such a domain temporal mobility occurs rather frequently.

Method	Kernel Type	Prec @10%	Prec @15%	Prec @20%
Using NC and TM features	Linear	100	100	90
	Radial basis	100	100	90
Using NC, TM and Graph Centrality features	Linear	100	100	90
	Radial basis	100	100	90
Method proposed in [10]	SP	46.7	51.7	46.7
	GL3	33.3	41.7	36.7
	GL4	46.7	41.7	44.4

Table 3: Precision of multi-node class @ Top-k(%) for DBLP dataset

Method	Kernel Type	Prec @10%	Prec @15%	Prec @20%
Using NC and TM features	Linear	100	100	90
	Radial basis	100	100	90
Using NC, TM and Graph Centrality features	Linear	100	100	90
	Radial basis	100	100	90
Method proposed in [10]	SP	66.7	54.1	60.0
	GL3	60.0	58.3	56.7
	GL4	46.7	47.5	46.7

Table 4: Precision of multi-node class @ Top-k(%) for Arnetminer dataset

## 5.2 Evaluation of Supervised Disambiguation

The main objective of our work is to find the  $s$ -value of a set of nodes in an unsupervised learning setup. These values can be used for the purpose of pre-filtering a small set of suspicious nodes which can be examined more thoroughly in a subsequent stage. However, we can also use our method in a supervised learning setup to predict whether an entity is a multi-node or not. For this, we use  $NC$ -score,  $TM$ -score, and network centrality based metrics as classification features and use SVM classification tool for classification. We use the LIBSVM library with default parameter setting. During the training phase, we use the  $-b$  option of this library to predict the probability instead of predicting the class label. This makes it easier to report the performance using AUC metric. While reporting accuracy, we simply predict the instances with a probability value higher than 0.50 as positive case (a multi-node), and the remaining as a negative case.

In Table 1 and 2, we show the accuracy and AUC value for both the datasets using a 10-fold cross validation for various kernels. As we can see, for both DBLP and Arnetminer, using these features, the best classification accuracy is achieved for the linear kernel and sigmoid kernel, which are 75.60% and 70.03%, respectively. For the case of AUC, all the kernels have almost similar performance, with the best value of 0.83, and 0.80 for DBLP and Arnetminer, respectively. Considering the fact that the method works on anonymized network, and only use topological features, accuracy value around 75% or AUC value around 0.80 are indeed commendable.

For this setup, we also report the precision@top- $k$  for  $k$  values equal to 10%, 15%, and 20% of the size of the test datasets. We use 3-fold cross validation for this experiment. To compute the above precisions, we simply sort the probability output of SVM in descending order and find the precision of the model in its desired range. The results are shown in Table 3 and Table 4 for the two datasets. We see that on DBLP dataset, all the top 15% of the probability values are more than 50%, thus they are predicted as positive (multi-node) class and in real-life all

those instances also belong to the true positive (multi-node) class, which yields a precision of 100%. For the case of top 20%, this value drops to 90%. The result on the Arnetminer dataset is also similar (see Table 4). This result shows that our method is able to place most of the true multi-nodes at the top part of its ranking table, as is desired.

For the supervised setting, we also report the results only based on NC and TM features in terms of accuracy, AUC and precision@top- $k$ . We can observe that adding centrality based features improves the results in terms of accuracy and AUC. As we can see, for both DBLP and Arnetminer, using only these two features, the best classification accuracy is achieved for the linear kernel and radial basis kernel, which are 72.50% and 68.82%, respectively. For the case of AUC, all the kernels have almost similar performance, with the best value of 0.80, and 0.76 for DBLP and Arnetminer, respectively. For precision@top- $k$  setup, the results of using NC and TM as classification features are almost the same compared with the results of adding centrality based graph topological features. Overall, the marginal improvement of using centrality based features are not that significant, which confirms that the *NC*-score, and the *TM*-score that we build are strong features for this classification task.

### 5.3 Comparison with existing works

The work by Hermansson et al. [10] is closely related to our work as they design a collection of graph kernels to classify multi-nodes in a supervised learning setup. Their kernels use only the graph topology, such as, graphlet counts and shortest paths, so they can be used in an anonymized network for entity disambiguation. To compare with their method, we run LIBSVM on our dataset using their best performing kernels, namely, size-3 graphlets (GL3), size-4 graphlets (GL4) and shortest path (SP) kernels. The kernel values are obtained by source code supplied by the authors. In Table 1, Table 2, Table 3 and Table 4, we compare the performance of our method that uses *NC*-score, *TM*-score, and centrality based graph topology as features with their method that uses topology based kernels, on all three performance metrics, accuracy, AUC, and precision@top- $k$ . As we can see our method performs much better than their method on these datasets. For instance, their best kernel achieves only 48.22% accuracy on DBLP and 47.67% accuracy on Arnetminer, whereas our method achieves 75.60% and 70.03% accuracy on DBLP, and Arnetminer. Cross-validation  $t$ -test shows that our method is significantly better ( $p$ -value 0.0051 for DBLP, and 0.0013 for Arnetminer). On AUC measure, our method obtains 0.83 and 0.80 on these datasets, whereas their method achieves a value of 0.64, and 0.62, which are much lower.

Besides improved performance, another advantage of our method over the methods in [10] is the superior running time. For a given node in the graph, the running time to compute the value of our features are only a few seconds, whereas computing graphlet kernel values is costly. In our experiments, for some of the nodes, the Matlab code provided by the authors of [10] took more than 2 days in a commodity PC.



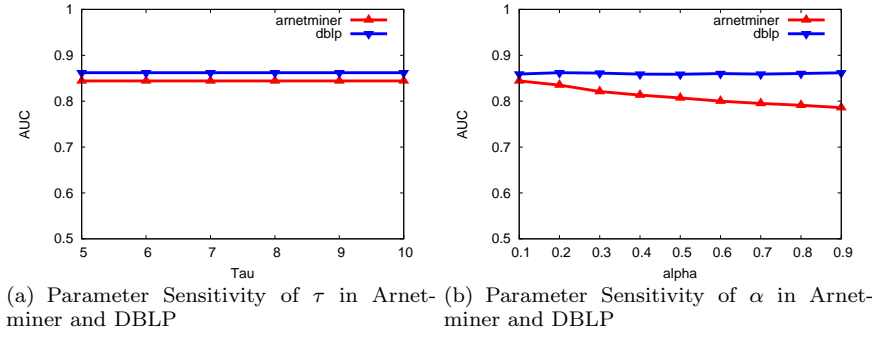


Fig. 4: Parameter Sensitivity Of Our Method On Two Datasets

#### 5.4 Study of Parameter Sensitivity

In case of unsupervised disambiguation task, our method uses two parameters that we set manually, one is the exponential decay rate ( $\tau$ ) for similarity computation, and the other is  $\alpha$  value in Equation 4. In this experiment, we see how the performance of the model changes as we vary the value of these parameters. The result of this experiment is shown in Figure 4(a) and Figure 4(b), where we plot the AUC value for a range of parameter values. From Figure 4(a) we see that the performance is very stable as we vary  $\tau$ . However, the performance degrades for the choice of  $\alpha$  but not that significantly.

#### 5.5 Study of Dataset Bias

Both our datasets are balanced, having equal number of positive and negative cases. However in real life scenario it would not be the characteristic of a wild dataset where the fraction of ambiguous entities is much lower. In this experiment we change the ratio of these two cases, to find the effect of dataset bias on the result quality. For this purpose, we change the size of positive instances and always keep the negative instances constant which is 75 negative

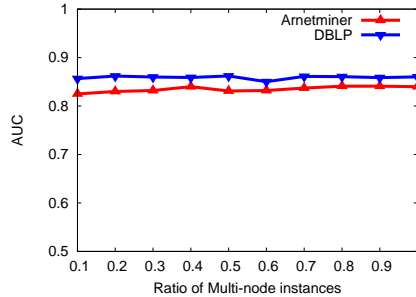


Fig. 5: Bias Effect in DBLP and Arnetminer

instances during the experiments. We randomly select part of positive instances and run the method ten times and get the mean AUC as final AUC value. The result of this experiment is shown in Figure 5; as we can see the performance varies for different ratios but not that significantly.

#### 5.6 Study of Running Time

A very desirable feature of our method is its running time. We have only two features and both of them can be calculated in a very short time. To compute the

Name	Number of direct neighbors	Running time (seconds)
Wei Wang	1375	2.85
Jiawei Han	523	0.55
Philip S. Yu	488	0.45
Wen Gao	531	0.50
Tao Li	603	0.58

Table 5: Running time result in DBLP

running time, we run our unsupervised disambiguation method on 5 entities from the DBLP datasets that have the largest number of neighbors. The running time on these vertices is shown in Table 5. The Arnetminer dataset is smaller than the DBLP datasets, so running time on the nodes of this dataset is even smaller.

Name	Ground Truth	DBLP probability	Arnetminer probability
Huan Liu	+	0.80	0.68
Tao Li	+	0.86	0.75
Wei Wang	+	0.87	0.77
Tao Xie	+	0.83	0.71
Bin Li	+	0.86	0.75
<b>Robert Allen</b>	+	<b>0.37</b>	<b>0.23</b>
Tim Weninger	-	3.2e-07	0.02
Jianlin Cheng	-	5.8e-11	0.00072
Hector Gonzalez	-	7.4e-06	0.02
Xifeng Yan	-	0.38	0.42
<b>Philip S. Yu</b>	-	<b>0.80</b>	<b>0.70</b>

Table 6: Real-life Case Study showing prominent researchers in DBLP and Arnetminer datasets

### 5.7 Real-life Case Study

In Table 6, we show the performance of our method on some of the well-known researchers from data mining and information retrieval communities. For each of the researchers, we denote the ground truth in the second column of the table. A positive sign stands for the fact that in DBLP and Arnetminer datasets the publication records under their names correspond to more than one real-life entity, and vice-versa. In the same table we also show the probability value that we obtain by our supervised disambiguation experiment that we discussed in Section 5.2. As we can see for many well known cases of multi-nodes in DBLP, such as Wei Wang, Huan Liu and Tao Li, our method correctly predicts their labels. A significant mistake (the mistaken cases are shown in bold fonts) that it makes is that it also predicts Professor Philip S. Yu to be a multi-node. This is a case of false positive, which our method is more susceptible. The reason for it is that many researchers have multiple disjoint communities that they maintain concurrently, so for such a researcher the *NC*-score is relatively small; also since her clusters do not exhibit temporal mobility, the *TM*-score for her case is also small. So, our method tends to predict such a person as positive. On the other hand false negative occurs in our method due to the fact that the *TM*-score undesirably improves the overall

score of a true positive case, even though the  $NC$ -score of that case is very small. One such example is Robert Allen as we show in this table.

## 6 Discussion and Conclusion

In this paper, we propose a novel solution to the entity disambiguation task in an anonymized network. We discuss the motivation of this task and show that our solution is useful for solving the entity disambiguation task in a constrained setting, where biographical features of the actors are not available. We also discuss how our solution can be used to find a small set of suspects for whom more detailed analysis can be made in a follow-through process. Another key strength of our method is that it is robust and it uses a simple model having only two features, normalized-cut score and temporal mobility score. Nevertheless, experiments on academic collaboration networks show that our method have excellent performance. Interestingly, for these datasets, temporal mobility score improves the prediction performance significantly. We believe that the dramatic improvement using temporal mobility feature on these datasets is due to the fact that in academic domain temporal mobility occurs rather frequently. However, due to the unavailability of ground truth datasets, we could not study whether this phenomenon presents in other networks, such as Phone call or online social networks, like Facebook. So we do acknowledge that the validity of our current work is particularly linked to academic collaboration networks and we leave the generalization of this work to networks from other domains as a future research direction.

## References

1. Allison, P., Long, J.S.: Interuniversity mobility of academic scientists. *American Sociological Rev.* **52**(5), 643–652 (1987)
2. Bhattacharya, I., Getoor, L.: Deduplication and group detection using links. In: *ACM SIGKDD Workshop on Link Analysis and Group Detection (LinkKDD)* (2004)
3. Bhattacharya, I., Getoor, L.: A latent dirichlet model for unsupervised entity resolution. In: *SDM* (2006)
4. Bunescu, R., Pasca, M.: Using encyclopedic knowledge for named entity disambiguation. In: *Proc. of the 11th Conf. of the European Chapter of the Association for Comp. Linguistics*, pp. 9–16 (2006)
5. Cen, L., Dragut, E.C., Si, L., Ouzzani, M.: Author disambiguation by hierarchical agglomerative clustering with adaptive stopping criterion. In: *SIGIR*, pp. 741–744 (2013)
6. Chin, W.S., Juan, Y.C., et al.: Effective string processing and matching for author disambiguation. In: *Proc. of the 2013 KDD Cup 2013 Workshop*, pp. 7:1–7:9 (2013)
7. Cucerzan, S.: Large-scale named entity disambiguation based on Wikipedia data. In: *Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 708–716 (2007)
8. Han, H., Giles, L., Zha, H., Li, C., Tsioutsoulis, K.: Two supervised learning approaches for name disambiguation in author citations. In: *Proc. of the 4th ACM/IEEE-CS Joint Conf. on Digital Libraries* (2004)
9. Han, H., Zha, H., Giles, C.L.: Name disambiguation in author citations using a k-way spectral clustering method. In: *Proc. of the 5th ACM/IEEE-CS Joint Conf. on Digital Libraries*, pp. 334–343 (2005)
10. Hermansson, L., Kerola, T., Johansson, F., Jethava, V., Dubhashi, D.: Entity disambiguation in anonymized graphs using graph kernels. In: *Proc. of the 22nd ACM international conference on information knowledge management*, pp. 1037–1046 (2013)
11. Jackson, M.O.: *Social and Economic Networks*. Princeton, NJ (2008)
12. Jackson, M.O.: *Social and Economic Networks*. Princeton University Press, Princeton, NJ, USA (2008)

13. Kataria, S.S., Kumar, K.S., Rastogi, R.R., Sen, P., Sengamedu, S.H.: Entity disambiguation with hierarchical topic models. In: Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1037–1045 (2011)
14. Li, Y., Wang, C., Han, F., Han, J., Roth, D., Yan, X.: Mining evidences for named entity disambiguation. In: Proc. of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1070–1078 (2013)
15. Liu, J., Lei, K.H., Liu, J.Y., Wang, C., Han, J.: Ranking-based name matching for author disambiguation in bibliographic data. In: Proc. of the 2013 KDD Cup 2013 Workshop, pp. 8:1–8:8 (2013)
16. von Luxburg, U.: A tutorial on spectral clustering. CoRR **abs/0711.0189** (2007)
17. Malin, B.: Unsupervised name disambiguation via social network similarity. In: In Proc. of the SIAM Workshop on Link Analysis, Counterterrorism, and Security, pp. 93–102 (2005)
18. Minkov, E., Cohen, W.W., Ng, A.Y.: Contextual search and name disambiguation in email using graphs. In: Proc. of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 27–34 (2006)
19. Newman, M.E.J.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences **103**(23), 8577–8582 (2006). DOI 10.1073/pnas.0601602103. URL <http://www.pnas.org/content/103/23/8577.abstract>
20. Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02, pp. 269–278. ACM, New York, NY, USA (2002). DOI 10.1145/775047.775087. URL <http://doi.acm.org/10.1145/775047.775087>
21. Sen, P.: Collective context-aware topic models for entity disambiguation. In: Proc. of the 21st Intl. Conference on WWW, pp. 729–738 (2012)
22. Tan, Y.F., Kan, M.Y., Lee, D.: Search engine driven author disambiguation. In: Proc. of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 314–315 (2006)
23. Tang, J., Fong, A.C.M., Wang, B., Zhang, J.: A unified probabilistic framework for name disambiguation in digital library. IEEE Trans. on Knowl. and Data Eng. **24**(6), 975–987 (2012)
24. Van Dongen, S.: Graph clustering via a discrete uncoupling process. SIAM J. Matrix Anal. Appl. **30**(1), 121–141 (2008)
25. Wang, F., Li, J., Tang, J., Zhang, J., Wang, K.: Name disambiguation using atomic clusters. In: Proc. of the 2008 The Ninth International Conference on Web-Age Information Management, pp. 357–364 (2008)
26. Wang, X., Tang, J., Cheng, H., Yu, P.S.: Adana: Active name disambiguation. In: Proc. of the 2011 IEEE 11th International Conference on Data Mining, pp. 794–803 (2011)
27. Whang, S.E., Garcia-Molina, H.: Entity resolution with evolving rules. Proc. VLDB Endow. **3**(1-2), 1326–1337 (2010). DOI 10.14778/1920841.1921004. URL <http://dx.doi.org/10.14778/1920841.1921004>
28. Whang, S.E., Menestrina, D., Koutrika, G., Theobald, M., Garcia-Molina, H.: Entity resolution with iterative blocking. In: SIGMOD 2009. Stanford (2009). URL <http://ilpubs.stanford.edu:8090/915/>
29. Yin, X., Han, J., Yu, P.: Object distinction: Distinguishing objects with identical names. In: Data Engineering, pp. 1242–1246 (2007)
30. Zhang, B., Saha, T.K., Hasan, M.A.: Name disambiguation from link data in a collaboration graph. In: 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2014, Beijing, China, August 17–20, 2014, pp. 81–84 (2014). DOI 10.1109/ASONAM.2014.6921563. URL <http://dx.doi.org/10.1109/ASONAM.2014.6921563>
31. Zhang, D., Tang, J., Li, J., Wang, K.: A constraint-based probabilistic framework for name disambiguation. In: Proc. of the Sixteenth ACM Conf. on Information and Knowledge Management, pp. 1019–22 (2007)